Title
Economic MPC of Nonlinear Processes via Recurrent Neural Networks Using Structural Process Knowledge

Author
Park, Michael Jihyuck

Publication Date
2020

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Economic MPC of Nonlinear Processes via Recurrent Neural Networks Using Structural Process Knowledge

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science
in Chemical Engineering

by

Michael Jihyuck Park

2020

ABSTRACT OF THE THESIS

Economic MPC of Nonlinear Processes via Recurrent Neural Networks Using Structural Process Knowledge

by

Michael Jihyuck Park

Master of Science in Chemical Engineering

University of California, Los Angeles, 2020

Professor Panagiotis D. Christofides, Chair

This work discusses three methods that incorporate a priori process knowledge into recurrent neural network (RNN) modeling of nonlinear processes to get increased prediction accuracy and provide information on how the neural network models are structured. The first method proposes a hybrid model that integrates first-principles models and RNN models together. The second method proposes a partially-connected RNN model which its structure is based on a priori structural process knowledge. The third method proposes a weight-constrained RNN model that integrates weight constraints into the training of the RNN model. The proposed RNN models are used in an economic model predictive control system and then applied to a chemical process example to validate the improved approximation performance compared to a fully-connected RNN model that is treated as a black box model.

The thesis of Michael Jihyuck Park is approved.

Dante A. Simonetti

Carlos Gilber Morales Guio

Panagiotis D. Christofides, Committee Chair

University of California, Los Angeles

2020

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

Thank you to Professor Christofides, and everyone in the Christofides lab, past and present, for the guidance and support since the start of my Master's Program. I have learned a lot from everyone.

Thank you to Professor Simonetti and Professor Guio for being on my committee.

This work was submitted with the same title for the presentation in the International Federation of Automatic Control (IFAC) World Congress 2020, and is co-authored by Zhe Wu, David Rincon, and Panagiotis D. Christofides. I would like to thank them for their contributions and for guiding me throughout the course of the thesis.

Chapter 1

Introduction

Machine learning has been widely adopted in many applications in chemical engineering processes in which neural networks have played a key role in modeling nonlinear systems [9, 14]. As recurrent neural networks (RNN) are able to capture temporal dynamic behavior, they have been utilized to model nonlinear dynamic systems and have been incorporated in the design of model-based predictive controllers that optimize process performance based on RNN prediction results [17, 18]. Using model predictive control (MPC) in a chemical process scenario allows the optimization process to operate in real-time, although an accurate process model must be used to be able to predict states. The process model can be derived from a first-principles model based on physical knowledge or a data-driven model based on industrial/simulation data [18]. It is noted that neural network modeling is generally treated as a black-box modeling where no physical knowledge is introduced. While the 'black-box' characteristics make it easy to implement, interpretability and optimality of neural network network modeling remain questionable.

On the other hand, chemical processes have been studied for a long time by researchers and engineers, where first-principles knowledge is obtained based on their predefined and well-known structure. For example, a chemical plant is designed in a sequence of intricate operation units that perform reactions, separations, among many other operations in which raw materials are fed in the first unit and products are obtained in the last unit in its simplest structure. Additionally,

it is also very common that some processes are highly coupled among units through reflux of unreacted material that is recycled to upstream units to maximize the production [8, 13]. However, the structural information of chemical plants is not utilized at any point during the training process of fully-connected neural network models that treat the plant as a black box.

While the fully-connected neural networks are developed based on the assumption that all the inputs affect all the neural network neurons, followed by all the outputs, in realistic chemical processes, it is possible that only a portion of inputs affect a portion of outputs. In order to make better use of such a priori process knowledge, many researchers have started to incorporate physical knowledge of systems in the neural network formulation (e.g., [1–3, 6, 7, 10]). Recently, a neural network has been specialized by including partial physical knowledge in its structure in [7]. In this paper, the nodes of the first layer represent the variables with physical meaning and the connection with the inputs are based on the impact between them. It was demonstrated that the resulting neural networks were able to improve the performance when compared with a fully-connected network.

Motivated by the above considerations, in this work, we propose a hybrid model, a partially-connected RNN model, and a weight-constrained RNN model to incorporate the physical knowledge into RNN modeling and training. The partially-connected RNN model and the weight-constrained RNN model are applied in the context of economic model predictive control (EMPC) of a chemical process example to demonstrate their benefits over the fully-connected RNN model.

Chapter 2

Preliminaries

2.1 Notation

The Euclidean norm of a vector is denoted by the operator $|\cdot|$ and the weighted Euclidean norm of a vector is denoted by the operator $|\cdot|_Q$ where Q is a positive definite matrix. x^T denotes the transpose of x. The notation $L_f V(x)$ denotes the standard Lie derivative $L_f V(x) := \frac{\partial V(x)}{\partial x} f(x)$. Set subtraction is denoted by "\", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$.

2.2 Class of Systems

The class of continuous-time nonlinear systems considered is described by the following state-space form:

$$\dot{x} = F(x,u) := f(x) + g(x)u,\ x(t_0) = x_0 \tag{2.1}$$

where $x \in \mathbf{R}^n$ is the state vector, and $u \in \mathbf{R}^m$ is the manipulated input vector. The control action constraint is defined by $u \in U := \{u_{min} \leq u \leq u_{max}\} \subset \mathbf{R}^m$, where u_{min} and u_{max} represent the minimum and the maximum value vectors of inputs allowed, respectively. $f(\cdot)$ and $g(\cdot)$ are sufficiently smooth vector and matrix functions of dimensions $n \times 1$ and $n \times m$, respectively. Without loss of generality, the initial time t_0 is taken to be zero $(t_0 = 0)$, and it is assumed that

$f(0) = 0$, and thus, the origin is a steady-state of the system of Eq. 2.1.

2.3 Stabilizability Assumptions via Lyapunov-based Control

We assume that there exists a positive definite Control Lyapunov function (CLF) V for the nonlinear system of Eq. 2.1 that satisfies the small control property (i.e., for every $\varepsilon > 0$, $\exists \delta > 0$, s.t. $\forall x \in \mathscr{B}_\delta(0)$, there exists u that satisfies $|u| < \varepsilon$ and $L_f V(x) + L_g V(x)u < 0$, [11]) and the following condition:

$$L_f V(x) < 0, \forall x \in \{z \in \mathbf{R}^n \backslash \{0\} \mid L_g V(z) = 0\} \tag{2.2}$$

The CLF assumption implies that there exists a stabilizing feedback control law $\Phi(x) \in U$ for the system of Eq. 2.1 that renders the origin of the closed-loop system asymptotically stable for all x in a neighborhood of the origin in the sense that $L_f V(x) + L_g V(x)u < 0$ holds for $u = \Phi(x) \in U$. An example of a feedback control law can be found in [5]. Based on the CLF assumption, we can first characterize a region where the time-derivative of V is rendered negative under the controller $\Phi(x) \in U$ as $\phi_u = \{x \in \mathbf{R}^n \mid \dot{V}(x) = L_f V + L_g V u < -kV(x), u = \Phi(x) \in U\} \cup \{0\}$, where k is a positive real number. Then, we define the closed-loop stability region Ω_ρ for the nonlinear system of Eq. 2.1 as a level set of the Lyapunov function embedded in $\phi_u : \Omega_\rho := \{x \in \phi_u \mid V(x) \leq \rho\} \subset \phi_u$, where $\rho > 0$.

2.4 Recurrent Neural Network Model

The following recurrent neural network (RNN) model is developed to approximate the nonlinear system of the Eq. 2.1 within the stability region Ω_ρ:

$$\dot{\hat{x}} = F_{nn}(\hat{x}, u) := A\hat{x} + \Theta^T y \tag{2.3}$$

where $\hat{x} \in \mathbf{R}^n$ is the RNN state vector and $u \in \mathbf{R}^m$ is the manipulated input vector. $y = [y_1, \ldots, y_n, y_{n+1}, \ldots, y_{m+n}] = [\sigma(\hat{x}_1), \ldots, \sigma(\hat{x}_n), u_1, \ldots, u_m] \in \mathbf{R}^{n+m}$ is a vector of both the network state $\hat{x}$ and the input u, where $\sigma(\cdot)$ is the nonlinear activation function (e.g., a sigmoid function $\sigma(x) = 1/(1+e^{-x})$). A is a diagonal coefficient matrix, i.e., $A = diag\{-a_1, \ldots, -a_n\} \in \mathbf{R}^{n \times n}$ with $a_i > 0$, and $\Theta = [\theta_1, \ldots, \theta_n] \in \mathbf{R}^{(m+n) \times n}$ with $\theta_i = b_i[w_{i1}, \ldots, w_{i(m+n)}], i = 1, \ldots, n$, where a_i and b_i are constants. w_{ij} is the weight connecting the jth input to the ith neuron where $i = 1, \ldots, n$ and $j = 1, \ldots, (m+n)$. The development of RNN models for the nonlinear system of Eq. 2.1 follows the data collection, training, and testing processes. Although an RNN model is able to approximate any complex nonlinear systems according to the universal approximation theorem [4, 12], how to obtain the optimal weight for RNN modeling of a nonlinear system remains challenging due to algorithmic learnability, complexity of neural network structure, and availability of computing power. In this work, we propose several approaches to optimizing neural network structure by incorporating physical knowledge into neural network design.

Chapter 3

Domain Adapted RNN

In this chapter, we introduce three different methods to integrate domain knowledge into neural network modeling and training. Specifically, instead of treating the RNN system of Eq. 2.3 like a black box model and training it using all the inputs and outputs available (termed the fully-connected model throughout the manuscript), we incorporate the process knowledge of the nonlinear system of Eq. 2.1 into RNN structure. The first method is to develop a hybrid model that integrates first-principles models with RNN models. The second method is to develop a partially-connected RNN structure using a priori knowledge of process input-output relationship. Lastly, a weight-constrained RNN model is developed by imposing constraints on the neural network weights based on the input-output relationship of the nonlinear system of Eq. 2.1.

3.1 Hybrid Model

While first-principles modeling has been studied and applied to chemical processes for over a century and has achieved good performances, it becomes difficult to obtain a 100% accurate first-principles model for large-scale systems due to inherent complexity. Therefore, in this work, we first propose a hybrid modeling method that introduces physical knowledge (e.g., first-principles knowledge based on physical laws such as mass and energy balances) into neural network modeling by combining a first-principles model and an RNN model together. Specifically,

the hybrid model is developed using an RNN function $\tilde{f}_{nn}(x,u)$ to approximate the gap between the first-principles model and the actual nonlinear process as follows:

$$\dot{x} = \tilde{f}(x) + \tilde{g}(x)u + \tilde{f}_{nn}(x,u) \tag{3.1}$$

where $\dot{x} = \tilde{f}(x) + \tilde{g}(x)u$ is the first-principles model that is developed based on general physical laws and assumptions, and therefore, may not be able to fully capture the dynamics of the actual nonlinear processes of Eq. 2.1 due to mismatch between $\dot{x} = \tilde{f}(x) + \tilde{g}(x)u$ and $\dot{x} = f(x) + g(x)u$. The RNN function $\tilde{f}_{nn}(x,u)$ in Eq. 3.1 is utilized to bridge the gap between the first-principles knowledge and the real process data. It is demonstrated that the hybrid model of Eq. 3.1 has the following advantages compared with a fully-connected RNN model. First, the RNN in the hybrid model is only used to approximate the residual between first-principles models and real process data, and therefore, may take less computing power and training time to learn. Additionally, when it comes to the operating region with no data available, the hybrid model can still be considered a reliable model due to its intrinsically physical knowledge, while the pure RNN model may be completely dysfunctional. In [19], a hybrid neural network model was developed for a chemical process where the linear part of the hybrid model is developed based on first-principles knowledge and the nonlinear term of reaction rate is provided by a neural network model using experiment/simulation data.

3.2 Partially-connected RNN Structure

We consider a case where the unit operations in the upstream stage of the production process affect those in the downstream stage in a chemical process, while the impact is ignorable in the opposite direction. To represent this relationship in mathematical form, we write the state vector and the input vector for the nonlinear system of Eq. 2.1 in the form of $x = [x^1, x^2] \in \mathbf{R}^n$ and $u = [u^1, u^2] \in \mathbf{R}^m$, and assume that the state vector x^1 is affected by u^1 only, and x^2 is affected by both u^1 and u^2. In Fig. 3.1, the fully-connected RNN model (left) is 'decoupled' to a partially-connected RNN

(right), from which it is demonstrated that input-output relationship of the partially-connected RNN is consistent with the above assumption. It is shown that the input vector u^1 is fed into the first RNN hidden layer and generates the output vector prediction of x^1. Then, the second input vector u^1 is combined with x^1 and then fed into the second RNN hidden layer, and generates the predicted output of x^2. The training process of a partially-connected RNN follows the same process for a fully-connected RNN model.

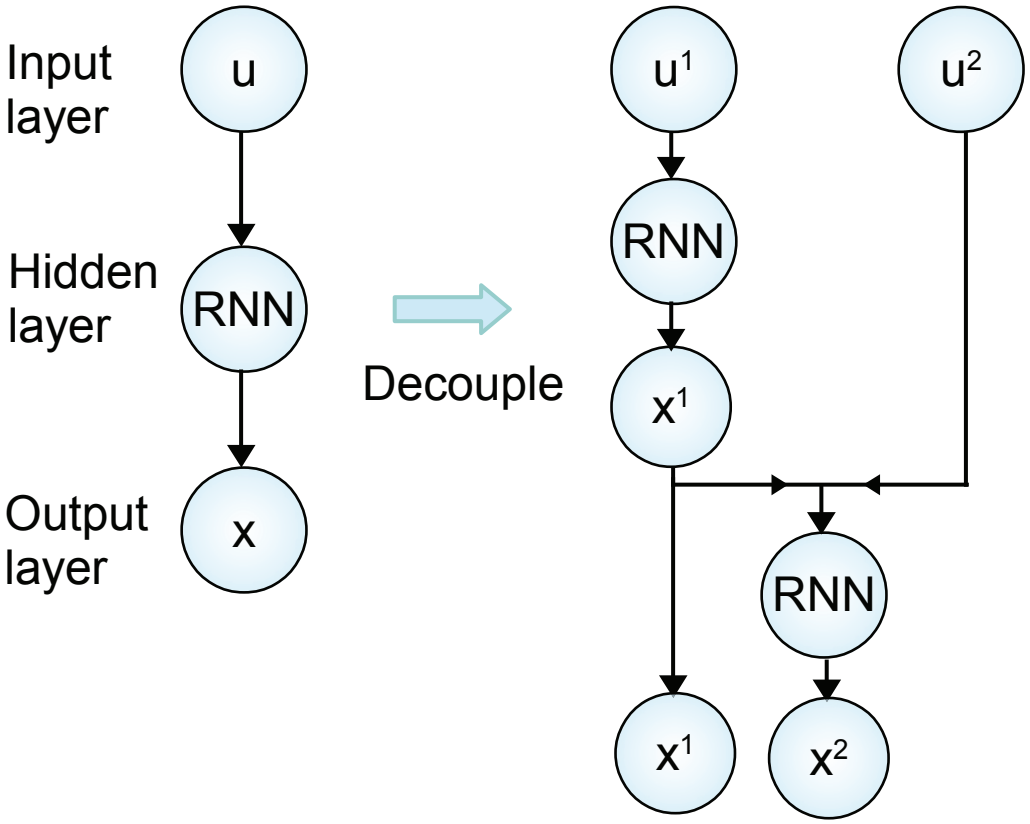

Figure 3.1: A partially-connected recurrent neural network structure based on process structural knowledge, where $u = [u^1, u^2]$ and $x = [x^1, x^2]$.

As the RNN structure is modified to infuse a priori knowledge on process structure into RNN modeling of the nonlinear system of Eq. 2.1, the training performance of RNN is expected to improve in terms of less computation time and higher prediction accuracy. Additionally, compared to a fully-connected model that takes all available inputs and outputs, less training data or fewer hidden neurons may be needed by the partially-connected RNN model to obtain a comparable

performance.

3.3 Weight-constrained RNN

Under the assumption that a portion of the input vector u^2 in the nonlinear system of Eq. 2.1 does not affect the output vector x^1, we develop a weight-constrained RNN model shown in Fig. 3.2 to eliminate the impact of process input u^2 on the state x^1. Specifically, to fully remove the connections between u^2 and x^1, another set of neurons $r_{h+1},\ldots,r_{2h}$ are added in the hidden layer as shown in Fig. 3.2. It is demonstrated that u^2 is disconnected from the neurons $r_1,\ldots,r_h$ that contribute to the output vector x^1 to eliminate the impact of u^2 on x^1. As a result, to maintain the impact of inputs on the other output vector x^2, the new set of neurons $r_{h+1},\ldots,r_{2h}$ are utilized in the hidden layer to connect both the inputs u^1 and u^2 to the output x^2. It is noted that compared to a fully-connected RNN model, the number of neurons and the number of weights in the weight-constrained RNN shown in Fig. 3.2 are increased to separate the connections to multiple output vectors.

Based on the RNN model of Eq. 2.3, the output vector x and the hidden neuron $r_i, i = 1,\ldots,2h$ in Fig. 3.2 are derived as follows:

$$\dot{x}^1 = \sum_{i=1}^{h} w_i^{(2)} \dot{r}_i, \quad \dot{x}^2 = \sum_{i=1}^{2h} w_i^{(2)} \dot{r}_i \tag{3.2}$$

$$\dot{r}_i = -a_i r_i + \theta_i y, \quad i = 1\ldots.,2h \tag{3.3}$$

where $\theta_i = b_i[w_{1i}^{(1)},\ldots,w_{(2h)i}^{(1)},\ldots,w_{(2h+m)i}^{(1)}]$ and $y = [\sigma(r_1),\ldots,\sigma(r_{2h}),u^1,u^2]^T$. a_i and b_i are constants, $w_{ji}^{(1)}$ is the weight connecting the jth input, $j = 1,\ldots,2h+m$ to the ith neuron, $i = 1,\ldots,2h$, and y is the input vector consisting of the hidden states r and the manipulated inputs u. $w^{(1)}$, $w^{(2)}$ represent the weight vectors before and after the hidden layer. Specifically, to train the weight-constrained RNN model with the structure of Fig. 3.2, we first develop a fully connected RNN model and then let the weights between u^2 and r_i, $i = 1,\ldots,h$, and the weights between r_i,

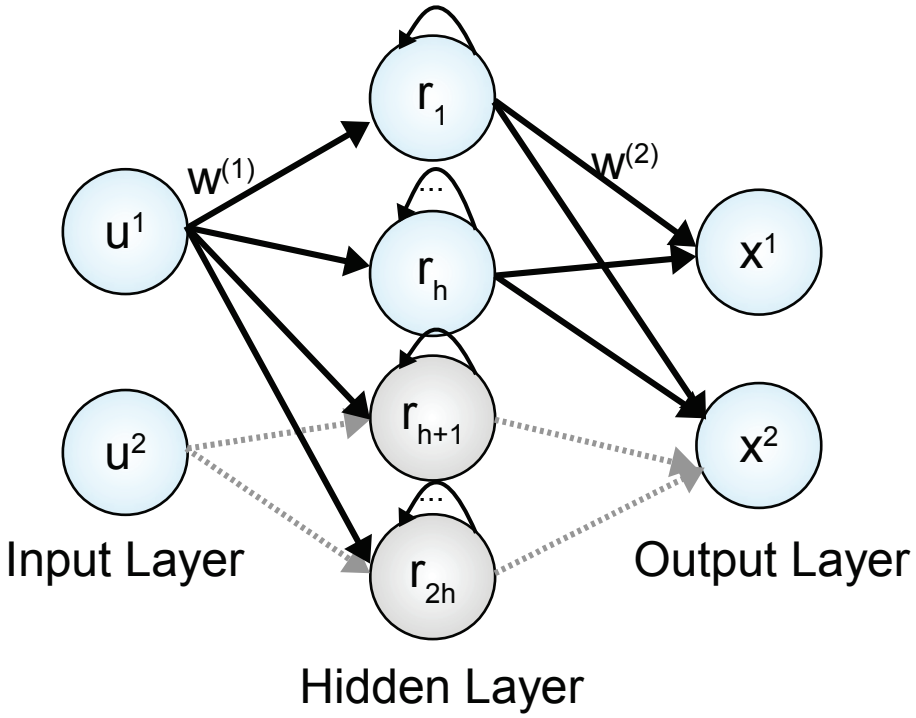

Figure 3.2: A recurrent neural network structure, where the connection between u^2 and x^1 is fully removed from the blue neurons, and the connection between u^2 and x^2 is rebuilt using the gray neurons in the hidden layer.

$i = h+1, \ldots, 2h$ and x^1 (denoted by $\tilde{w}$) be zero or constrained by a sufficiently small bound. Such constraints on th RNN weights need to be well-defined before training. It should be noted that since there exist three types of weight matrices in an RNN model: 1) the weight matrix connecting the input layer and the hidden layer, 2) the weight matrix feeding the past neuron information into the current network (i.e., the feedback loop in r_i, $i = 1, \ldots, 2h$), and 3) the weight matrix connecting the hidden layer to the output layer, the constraints need to be implemented in all the three weight matrices such that u^2 and x^1 can be fully disconnected.

In this work, we train the above weight-constrained RNN model in Keras, an open-source neural network library in Python, and implement weight constraints in the *constraints.py* source file. To develop an RNN model that obtains the optimal weights subject to the weight constraints, the RNN optimizer (e.g., *adaptive learning rate optimization algorithm*) needs to be modified to minimize loss function while accounting for the weight constraints in the optimization problem.

Alternatively, the weight constraints can be implemented at the end of each training epoch such that the weights that meet the constraints remain unchanged and those exceeding the constraints will be bounded to the saturation value. The saturated weights will then be utilized as the initial condition for the optimization problem for the next training epoch, and the above process is repeated until the stopping criteria of the training process are satisfied.

In addition to the constraints on RNN weights, penalty components on weight parameters can be employed in the loss function of the RNN optimization problem to introduce a priori weight knowledge into the training process. For example, regularization techniques (e.g., L1 and L2 regularization) can be utilized in the training process to obtain a less complex RNN model and reduce over-fitting. Specifically, the following loss function is developed to account for the weights $\tilde{w}$ that should be bounded in a weight-constrained RNN model:

$$L = \sum_{i=1}^{N_d} (x_i - \hat{x}_i)^2 + \lambda \sum_{i=1}^{h} |\tilde{w}| \tag{3.4}$$

where x_i and $\hat{x}_i$ are the actual and predicted outputs, respectively, N_d is the number of training data samples, and $\lambda > 0$ is the weight for the regularization term.

3.4 RNN Training Process and Stability

The RNN models in this work are developed using Keras library. Specifically, the hybrid model and the weight-constrained model are developed following the construction method for a fully-connected model, where the training dataset is preprocessed to represent the gap between the first-principles model and real process data, and the weight constraints are added into the constraints and optimizer files before training, respectively. To develop a partially-connected RNN model, an RNN layer is first developed to connect u^1 and x^1. Subsequently, x^1 and u^2 are concatenated and followed by a second RNN layer to ultimately obtain x^2. It is noted that instead of using the full input and output vectors u and x, the input vectors u^1, u^2 and the output vectors x^1, x^2 need to be specified and fed into the partially-connected RNN model separately.

Additionally, all the three RNN models are trained with a constraint on the modeling error (denoted by $|v| = |F(x,u,0) - F_{nn}(x,u)|$) such that the obtained RNN model can represent the actual process well and can be utilized in a model-based predictive controller that stabilizes the system at its steady-state with guaranteed stability.

Remark 3.1. *Consider the nonlinear system of Eq. 2.1 with $x = [x^1, x^2] \in \boldsymbol{R}^n$ and $u = [u^1, u^2] \in \boldsymbol{R}^n$, where x^1 and x^2, u^1 and u^2 are of the same dimension, respectively (i.e., $x^1, x^2 \in \boldsymbol{R}^{\frac{n}{2}}$, $u^1, u^2 \in \boldsymbol{R}^{\frac{m}{2}}$). Under the assumption of the input-output relationship in this chapter, the total number of weights for a partially-connected RNN model with two hidden layers, where each hidden layer has h neurons, is calculated to be $\frac{3}{2}nh + mh + 2h^2$, while the total number of weights for a fully-connected RNN model with the same two hidden layers is $mh + 3h^2 + nh$ (the bias term is ignored in the comparison as it can be considered a constant input node). Since in most cases, the number of neurons is much greater than the number of inputs and states to achieve a desired approximation performance, the number of weights for a decoupled RNN model is significantly reduced due to the incorporation of process structural knowledge ($\frac{3}{2}nh + mh + 2h^2 << nh + mh + 3h^2$ when $h >> m, n$). However, it is noted that the number of weights in a weight-constrained model with the structure of Fig. 3.2 is increased compared to the fully-connected RNN model due to the new set of hidden neurons that are used to rebuild the connection between u^2 and x^2*

Remark 3.2. *It is noted that all the RNN models in this chapter are developed for the nominal system of Eq. 2.1 without disturbances. However, in the presence of time-varying disturbances, the RNN model that is trained for the nominal system may be dysfunctional in a modelbased predictive controller due to a considerable model mismatch. To that end, online update of RNN models can be employed to capture the nonlinear dynamics subject to disturbances using the most recent process measurement data.*

3.5 RNN-based Predictive Control

The Lyapunov-based economic predictive control (LEMPC) using the RNN model of Eq. 2.3 is utilized to optimize process economic performance while maintaining the closed-loop state of the nonlinear system of Eq. 2.1 in the stability region Ω_ρ. the LEMPC is formulated by the following optimization problem [16]:

$$\mathscr{J} = \max_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} l_e(\tilde{x}(t), u(t)) dt \tag{3.5a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{3.5b}$$

$$u(t) \in U, \forall t \in [t_k, t_{k+N}) \tag{3.5c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{3.5d}$$

$$V(\tilde{x}(t)) \leq \rho_e, \forall t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_e} \tag{3.5e}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi_{nn}(x(t_k))), \text{ if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_e} \tag{3.5f}$$

where $\tilde{x}$ is the predicted state trajectory, $S(\Delta)$ is the set of piecewise constant functions with period Δ, N is the number of sampling periods in the prediction horizon, and $\dot{V}(x,u)$ represents $\frac{\partial V(x)}{\partial x}(F_{nn}(x,u))$. The optimization problem of Eq. 3.5 maximized the objective function of Eq. 3.5a that integrates $l_e(\tilde{x}(t), u(t))$ over the prediction horizon subject to the constraints of Eqs. 3.5b-3.5f. Specifically, the constraint of Eq. 3.5b is the RNN model of Eq. 2.3 used for prediction. Eq. 3.5c defines the input constraints applied over the entire prediction horizon. Eq. 3.5d defines the initial condition $\tilde{x}(t_k)$ of Eq. 3.5b as the state measurement at $t = t_k$. The constraint of Eq. 3.5e maintains the predicted closed-loop states in Ω_{ρ_e} if $x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_e}$, where $\Omega_{\rho_e}, 0 < \rho_e < \rho$, is a level set of Lyapunov function that guarantees the boundedness of state in the closed-loop stability region Ω_ρ, accounting for the model mismatch between the RNN model of Eq. 3.5b and the nonlinear process of Eq. 2.1. On the other hand, if $x(t_k)$ leaves Ω_{ρ_e}, the contractive constraint of Eq. 3.5f will be activated to drive the state towards the origin within the next sampling period. It is demonstrated that the closed-loop state of the nonlinear system of Eq. 2.1 is bounded in the stability region Ω_ρ

for all times under the LEMPC of Eq. 3.5.

Chapter 4

Application to a Chemical Process Example

A chemical process example is utilized to demonstrate the application of the proposed RNN modeling with the incorporation of structural process knowledge. Specifically, two well-mixed, non-isothermal continuous stirred tank reactors (CSTR) are considered where an irreversible second-order exothermic reaction takes place in each reactor as shown in Fig. 4.1. The reaction transforms a reactant A to a product B $(A \rightarrow B)$. Each of the two reactors are fed with reactant material A with the inlet concentration C_{Aj0}, the inlet temperature T_{j0}, and the feed volumetric flow rate of the reactor F_{j0}, $j = 1,2$, where $j = 1$ denotes the first CSTR and $j = 2$ denotes the second CSTR. Each CSTR is equipped with a heating jacket that supplies/removes heat at a rate Q_j, $j = 1,2$. The CSTR dynamic models are described by the following material and energy balance equations:

$$\frac{dC_{A1}}{dt} = \frac{F_{10}}{V_1}(C_{A10} - C_{A1}) - k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{4.1a}$$

$$\frac{dT_1}{dt} = \frac{F_{10}}{V_1}(T_{10} - T_1) + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 + \frac{Q_1}{\rho_L C_p V_1} \tag{4.1b}$$

$$\frac{dC_{B1}}{dt} = -\frac{F_{10}}{V_1} C_{B1} + k_0 e^{\frac{-E}{RT_1}} C_{A1}^2 \tag{4.1c}$$

$$\frac{dC_{A2}}{dt} = \frac{F_{20}}{V_2} C_{A20} + \frac{F_{10}}{V_2} C_{A1} - \frac{F_{10} + F_{20}}{V_2} C_{A2} - k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 \tag{4.1d}$$

$$\frac{dT_2}{dt} = \frac{F_{20}}{V_2} T_{20} + \frac{F_{10}}{V_2} T_1 - \frac{F_{10} + F_{20}}{V_2} T_2 + \frac{-\Delta H}{\rho_L C_p} k_0 e^{\frac{-E}{RT_2}} C_{A2}^2 + \frac{Q_2}{\rho_L C_p V_2} \tag{4.1e}$$

$$\frac{dC_{B2}}{dt} = \frac{F_{10}}{V_2}C_{B1} - \frac{F_{10}+F_{20}}{V_2}C_{B2} + k_0 e^{\frac{-E}{RT_2}}C_{A2}^2 \tag{4.1f}$$

where C_{Aj}, V_j, T_j, and Q_j, $j = 1,2$ are the concentration of reactant A, the volume of the reacting liquid, the temperature, and the heat input rate in the first and second reactor, respectively. The reacting liquid has a constant density of ρ_L and a heat capacity of C_p for both reactors. ΔH, k_0, E, and R represent the enthalpy of reaction, pre-exponential constant, activation energy, and ideal gas constant, respectively, and are the same for both reactors. Process parameter values are listed in Table 4.1.

Table 4.1: Parameter values of the CSTR

$T_{10} = 300\ K$	$T_{20} = 300\ K$
$F_{10} = 5\ m^3/hr$	$F_{20} = 5\ m^3/hr$
$V_1 = 1\ m^3$	$V_2 = 1\ m^3$
$T_{1_s} = 402\ K$	$T_{2_s} = 402\ K$
$C_{A1_s} = 1.95\ kmol/m^3$	$C_{A2_s} = 1.95\ kmol/m^3$
$C_{A10_s} = 4\ kmol/m^3$	$C_{A20_s} = 4\ kmol/m^3$
$Q_{1_s} = 0.0\ kJ/hr$	$Q_{2_s} = 0.0\ kJ/hr$
$k_0 = 8.46 \times 10^6\ m^3/kmol \cdot hr$	$\Delta H = -1.15 \times 10^4\ kJ/kmol$
$C_p = 0.231\ kJ/kg \cdot K$	$R = 8.314\ kJ/kmol \cdot K$
$\rho_L = 1000\ kg/m^3$	$E = 5 \times 10^4\ kJ/kmol$

The manipulated inputs for both CSTRs are the inlet concentration of species A and the heat input rate, which are represented by the deviation variables $\Delta C_{Aj0} = C_{Aj0} - C_{Aj0_s}$, $\Delta Q_j = Q_j - Q_{j_s}$, $j = 1,2$, respectively. The manipulated inputs are bounded as follows: $|\Delta C_{Aj0}| \leq 3.5\ kmol/m^3$ and $|\Delta Q_j| \leq 5 \times 10^5\ kJ/hr$, $j = 1,2$. The states and the inputs of the closed-loop system are $x^T = [C_{A1} - C_{A1_s}\ \ T_1 - T_{1_s}\ \ C_{A2} - C_{A2_s}\ \ T_2 - T_{2_s}]$ and $u^T = [\Delta C_{A10}\ \ \Delta Q_1\ \ \Delta C_{A20}\ \ \Delta Q_2]$, respectively, where C_{A1_s}, C_{A2_s}, T_{1_s}, and T_{2_s} are the steady-state values of concentration A and temperature in the first and second reactors, such that the equilibrium point of the system is at the origin of the state-space.

The control objective of LEMPC is to maximize profit of both CSTR systems described in

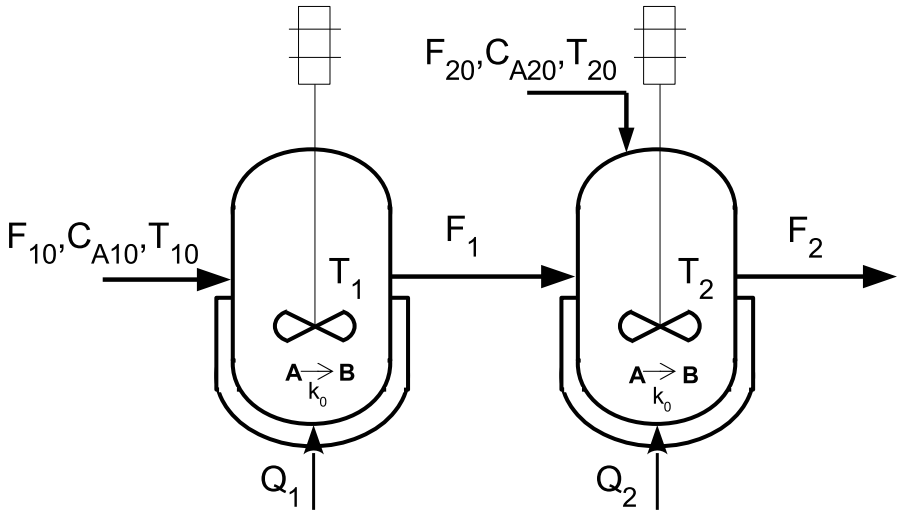

Figure 4.1: Process flow diagram of two CSTRs in series.

Eq. 4.1 by manipulating the inlet concentration ΔC_{A10} and C_{A20} and the heat inputs rate ΔQ_1 and ΔQ_2, and meanwhile maintain the closed-loop state trajectories in the stability region Ω_ρ for all times under LEMPC. The objective function of the LEMPC optimizes the production rate of B as follows:

$$l_e(\tilde{x},u) = k_0 e^{-E/RT_1} C_{A1}^2 + k_0 e^{-E/RT_2} C_{A2}^2 \tag{4.2}$$

The explicit Euler method with an integration time step of $h_c = 10^{-4}\ hr$ is used to numerically simulate the dynamic model of Eq. 4.1. The nonlinear optimization problem of LEMPC of Eq. 3.5 is solved using the python module of IPOPT software package [15], named PyIpopt with the sampling period $\Delta = 10^{-2}\ hr$. Two control Lyapunov functions $V_1(x) = x^T P_1 x$, and $V_2(x) = x^T P_2 x$

are designed for two CSTRs, respectively, with the following positive definite P matrices:

$$P_1 = P_2 = \begin{bmatrix} 1060 & 22 \\ 22 & 0.52 \end{bmatrix} \tag{4.3}$$

The closed-loop stability regions for the two CSTRs are characterized with $\rho = 380$ and $\rho_e = 260$. Following the general RNN development process in [18], a fully-connected RNN model, a partially-connected RNN model, and a weight-constrained RNN model are developed for the CSTR process of Eq. 4.1 using the same dataset with the same neural network parameters as follows: 2 hidden layers with 30 neurons in each layer, *tanh* as the activation function, and *Adam* as the optimizer.

Open-loop simulations are first carried out to demonstrate the open-loop prediction performances of the fully-connected RNN model, the partially-connected RNN model, and the weight-constrained RNN model, respectively. The mean square errors between the first-principles state trajectories (i.e., the state trajectories using the first-principles model of Eq. 4.1) and the above three models, respectively, are reported in Table 4.2, where P-RNN, W-RNN and F-RNN represent the partially-connected RNN model, the weight-constrained RNN model, and the fully-connected RNN model, respectively. From Table 4.2, it is demonstrated that the partially-connected RNN model and the weight-constrained RNN model outperform the fully-connected model in that the open-loop approximations of C_{A1}, C_{A2}, T_1, and T_2 are significantly improved.

Table 4.2: MSE comparison of open-loop prediction results with the first-principles model results.

	P-RNN	W-RNN	F-RNN
C_{A1} $(kmol/m^3)$	1.0×10^{-4}	5.6×10^{-6}	0.9×10^{-4}
T_1 (K)	0.14	0.018	0.15
C_{A2} $(kmol/m^3)$	8.2×10^{-7}	2.0×10^{-6}	2.6×10^{-6}
T_2 (K)	5.4×10^{-4}	0.0076	0.049

After demonstrating that all the three RNN models achieve desired prediction accuracy for

the CSTR process of Eq. 4.1 in the stability region, closed-loop simulations are performed under the LEMPC of Eq. 3.5 using the first-principles model of Eq. 4.1 and the three RNN models, respectively. In Fig. 4.2, it is demonstrated that the state trajectories for both CSTRs are bounded in the stability region Ω_ρ for all times under LEMPC. Fig. 4.3 shows the evolution the Lyapunov function values of V_1 and V_2 under LEMPC using the first-principles model of Eq. 4.1 and three different RNN models, respectively. Specifically, due to a relatively large model mismatch for the fully-connected RNN model as reported in Table 4.2, the contractive constraint of Eq. 3.5f is activated frequently under the LEMPC using a fully-connected RNN model because the actual process state does not stay in Ω_{ρ_e} under the constraint of Eq. 3.5e. As a result, it is observed in Fig. 4.3 that the V profiles under the fully-connected model show larger oscillation compared to those under the two RNN models and under the first-principles model.

Additionally, we compare the accumulated economic profits $L_E = \int_0^{t_p} L_e(x,u)dt$ within the operation period $t_p = 0.32\ hr$ for the closed-loop CSTRs under the steady-state operation (i.e., the CSTRs are operated at their steady-states for all times), and the LEMPC using the first-principles model of Eq. 4.1 and the three RNN models, respectively. The result is shown in Fig. 4.4, from which it is demonstrated that the closed-loop operation under LEMPC achieves higher economic profits than the steady-state operation. Specifically, the LEMPC using the first-principles model achieves the highest economic benefits since the closed-loop state trajectory reaches and stays at the boundary of Ω_{ρ_e} smoothly based on accurate predictions. Moreover, it is demonstrated that the LEMPC using the partially-connected RNN model and the weight-constrained RNN model economically outperform that under the fully-connected RNN model due to better prediction accuracy in the stability region. Therefore, through both open-loop and closed-loop simulations, we demonstrate that the domain-knowledge-based RNN models achieve desired approximation performance for the CSTR process of Eq. 4.1 and provide reliable state predictions for model-based predictive controllers.

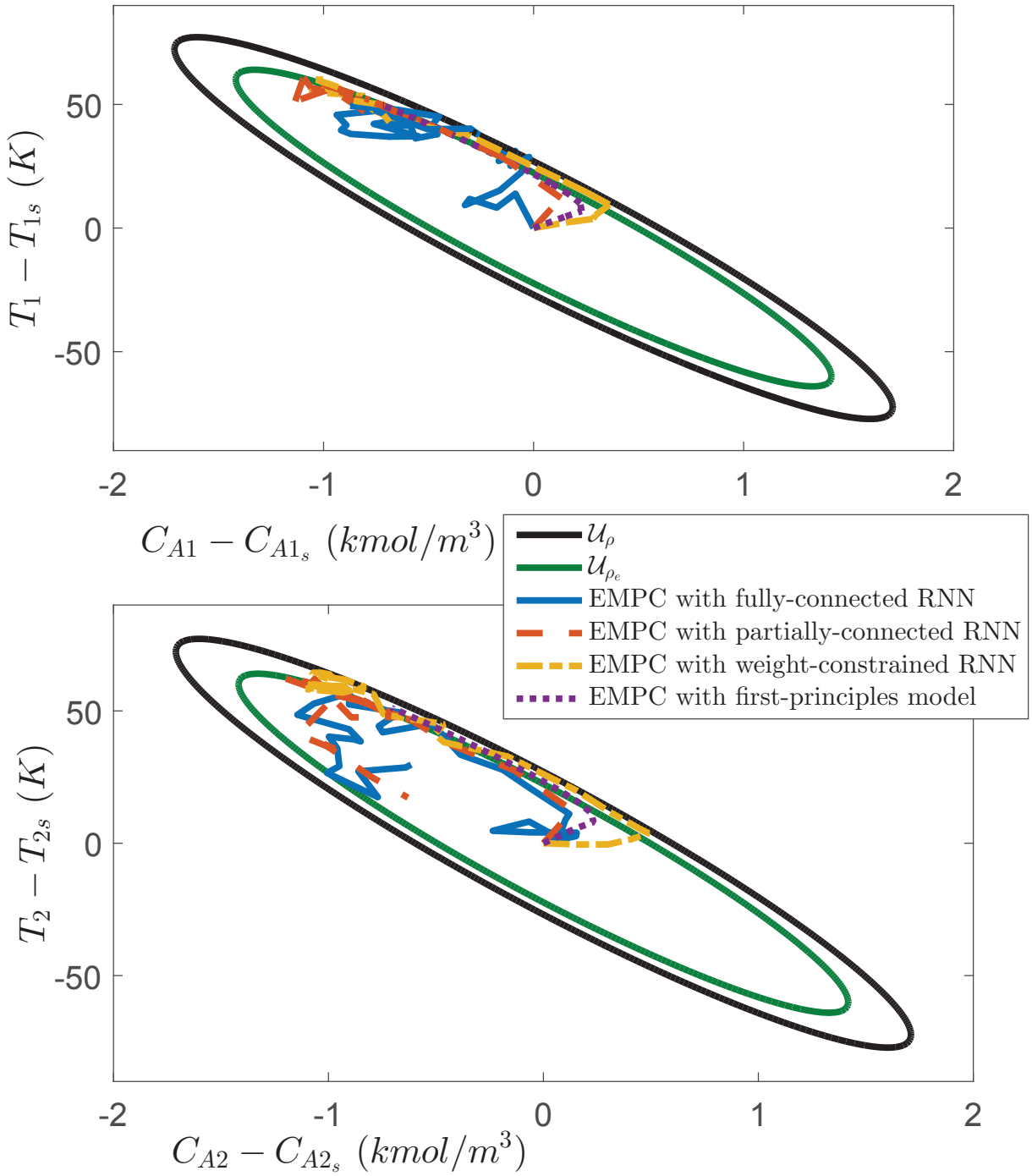

Figure 4.2: The state-space profiles for the closed-loop simulation for CSTR 1 (top) and CSTR 2 (bottom) under the EMPC using the fully-connected model, the partially-connected RNN model, the weight-constrained RNN model, and the first-principles model of Eq. 4.1, respectively, for an initial condition (0, 0, 0, 0).

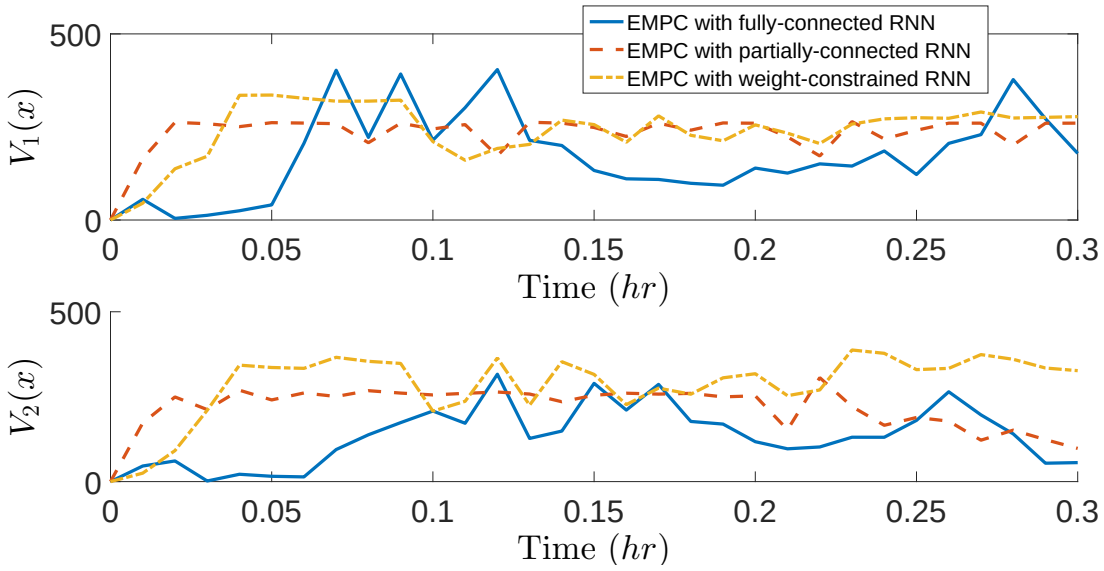

Figure 4.3: The Lyapunov function value with time for the closed-loop CSTR 1 (top) and CSTR 2 (bottom) under the EMPC using the fully-connected model, the partially-connected RNN model, the weight-constrained RNN model, and the first-principles model of Eq. 4.1, respectively, for an initial condition (0, 0, 0, 0).

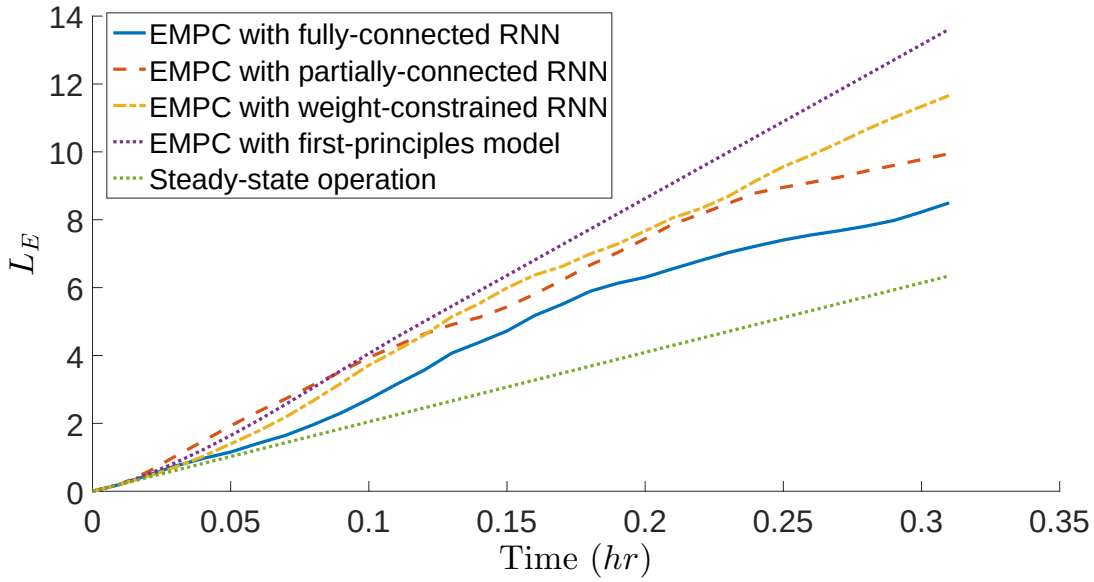

Figure 4.4: Accumulated economic profits for the closed-loop CSTR 1 (top) and CSTR 2 (bottom) under the steady-state operation and under the EMPC using the first-principles model of Eq. 4.1, the fully-connected model, the partially-connected RNN model, and the weight-constrained RNN model, respectively, for an initial condition (0, 0, 0, 0).

Chapter 5

Conclusion

In this work, we developed three modeling approaches that incorporates a priori process knowledge into RNN models. Specifically, a hybrid model that combines a first-principles model and an RNN model was first developed. Then, a partially-connected RNN model and a weight-constrained RNN model were developed based on an assumption on process input-output relationship. The partially-connected and weight-constrained RNN models were then applied to a chemical process example, from which it was demonstrated that the open-loop and closed-loop prediction performances under the LEMPC using the above two RNN models outperformed that under the fully-connected RNN model in terms of higher prediction accuracy, smoother state trajectories, and better economic benefits.

Bibliography

[1] Y. Ba, G. Zhao, and A. Kadambi. Blending diverse physical priors with neural networks. *arXiv preprint arXiv:1910.00201*, 2019.

[2] A. Karpatne, W. Watkins, J. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[3] M. Kellman, E. Bostan, N. Repina, and L. Waller. Physics-based learned design: Optimized coded-illumination for quantitative phase imaging. *IEEE Transactions on Computational Imaging*, 2019.

[4] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamical systems. *IEEE Transactions on Neural Networks*, 6:422–431, 1995.

[5] Y. Lin and E. D. Sontag. A universal formula for stabilization with bounded controls. *Systems & Control Letters*, 16:393–397, 1991.

[6] Z. Long, Y. Lu, X. Ma, and B. Dong. PDE-net: Learning PDEs from data. *arXiv preprint arXiv:1710.09668*, 2017.

[7] Y. Lu, M. Rajora, P. Zou, and S. Liang. Physics-embedded machine learning: case study with electrochemical micro-machining. *Machines*, 5(1):4, 2017.

[8] M. L. Luyben, B. D. Tyreus, and W. L. Luyben. Plantwide control design procedure. *AIChE Journal*, 43:3161–3174, 1997.

[9] S. J. Qin and L. H. Chiang. Advances and opportunities in machine learning for process data analytics. *Computers & Chemical Engineering*, 126:465–473, 2019.

[10] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S. Chung. Neural lander: Stable drone landing control using learned dynamics. In *Proceedings of the International Conference on Robotics and Automation*, pages 9784–9790, Montreal, Canada, 2019.

[11] E. D. Sontag. A 'universal' construction of artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13:117–123, 1989.

[12] E. D. Sontag. Neural nets as systems models and controllers. In *Proceedings of the Seventh Yale Workshop on Adaptive and Learning Systems*, pages 73–79, Yale University, 1992.

[13] R. Turton, R. C. Bailie, W. B. Whiting, and J. A. Shaeiwitz. *Analysis, synthesis and design of chemical processes*. Pearson Education, 2008.

[14] V. Venkatasubramanian. The promise of artificial intelligence in chemical engineering: Is it here, finally? *AIChE Journal*, 65:466–478, 2019.

[15] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.

[16] Z. Wu and P. D. Christofides. Economic machine-learning-based predictive control of nonlinear systems. *Mathematics*, 7(6):494, 2019.

[17] Z. Wu, D. Rincon, and P. D. Christofides. Real-time adaptive machine-learning-based predictive control of nonlinear processes. *Industrial & Engineering Chemistry Research*, 59:2275–2290, 2019.

[18] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. part I: Theory. *AIChE Journal*, 65:e16729, 2019.

[19] Z. Zhang, Z. Wu, D. Rincon, and P. D. Christofides. Real-time optimization and control of nonlinear processes using machine learning. *Mathematics*, 7(10):890, 2019.